Meta-domains for Autotmated System Indentification *

Matthew Easley
Elizabeth Bradley

CU-CS-904-00

University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

* Supported by NSF NYI #CCR-9357740, ONR #N00014-96-1-0720, and a Packard Fellowship in Science and Engineering from the David and Lucile Packard Foundation.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **2000**	2. REPORT TYPE	3. DATES COVERED **00-00-2000 to 00-00-2000**
4. TITLE AND SUBTITLE **Meta-domains for Automated System Indentification**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Department of Computer Science,University of Colorado,Boulder,CO,80309**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES		
14. ABSTRACT **see report**		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**18**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Meta-domains for Automated System Identification
Techincal Report #CU-CS-904-00
In review for *ANNIE00*

Matthew Easley and Elizabeth Bradley *
University of Colorado at Boulder
Department of Computer Science
Boulder, Colorado 80309–0430
{easley,lizb}@cs.colorado.edu

May 30, 2000

Abstract

We present a new knowledge representation and reasoning framework for modeling nonlinear dynamical systems. The goals of this framework are to smoothly incorporate varying levels of domain knowledge and to tailor the search space and the reasoning methods accordingly. In particular, we introduce a new structure for automated model building known as a *meta-domain* which, when instantiated with domain-specific components, tailors the space of candidate models to the system at hand. The `xmission-line` meta-domain, for instance, generalizes the notion of an electrical transmission line. It uses an iterative template to build models, and it may be easily customized into specific model-building domains, ranging from mechanical vibrations to thermal conduction. We combine this abstract modeling paradigm with ideas from generalized physical networks, a meta-level representation of idealized two-terminal elements, and a hierarchy of qualitative and quantitative analysis tools, to produce dynamic modeling domains whose complexity naturally adapts to the amount of available information about the target system.

*Supported by NSF NYI #CCR-9357740, ONR #N00014-96-1-0720, and a Packard Fellowship in Science and Engineering from the David and Lucile Packard Foundation.

1 Introduction

System identification (SID) is the process of identifying a dynamic model of an unknown system. The challenges involved in automating this process are significant, as applications in different fields of science and engineering demand different kinds of models and modeling techniques. System identification entails two steps, as shown in Figure 1: *structural identification*, wherein one ascertains

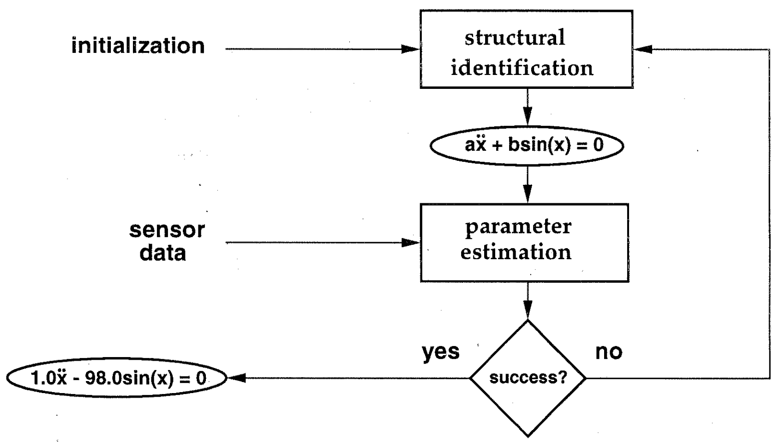

Figure 1: The system identification process

the general form of the model as described by an ordinary differential equation or ODE (e.g., $a\ddot{x} + b\sin(x) = 0$ for a simple pendulum), and then *parameter estimation*, in which one finds specific parameter values for the unknown coefficients that fit that model to observed data (e.g. $a = 1.0$, $b = -98.0$). For nonlinear systems, parameter estimation is difficult and structural identification is even harder; AI techniques can be used to automate the former[6], but the latter has, until now, remained the purview of human experts.

A central problem in any automated modeling task is that model complexity, and hence the size of the search space, is exponential in the number of model components unless severe restrictions are placed on the model-building process. A good compromise between *black-box* modeling, which uses no domain knowledge but has a prohibitively large search space, and *clear-box* modeling, where the modeler knows a great deal about the system, is *gray-box* modeling. Here, partial information about the internals of the box—e.g., whether the system is electronic or viscoelastic—is used to prune the search space to a reasonable size. The key to making gray-box modeling of nonlinear dynamical systems practical is a flexible knowledge representation scheme that adapts to the problem at hand.

The solution proposed in this paper combines a representation that allows for different levels of domain knowledge, a set of reasoning techniques that are appropriate to each level, and a control strategy that invokes the right technique at the right time. In particular, we introduce a new structure for automated model building known as a *meta-domain* which, when instantiated with domain components, tailors the model generation search space to a specific system identification task. To increase a meta-domain's applicability and utility, we also incorporate ideas from *generalized physical networks*[18], a meta-level representation of idealized two-terminal elements, traditional compositional model building[12], and qualitative reasoning[21]. The intent is to span the spectrum between highly specific frameworks that work well in a single, limited domain (e.g., a spring/dashpot vocabulary for modeling simple mechanical systems) and abstract frameworks that rely heavily upon general mathematical formalisms at the expense of having large search spaces. [7]).

2 Meta-domains for Model Building

Theoretically, an automated system identification tool could rely solely upon a brute-force model generator for its structural identification phase and a nonlinear parameter estimator for its testing phase. However, many implementation issues make this wholly impractical: parameter estimation is extremely expensive, and even simple application domains have an exponential number of valid models. Consider, for example, all of the permutations of a single electrical resistor, capacitor, and inductor connected in parallel and/or series. Although the complete set of permutations does describe all possible models that can be built with this set of components, the number of *functionally different* models in the set is significantly less.

The key to effective model building is to exploit this kind of knowledge to create an appropriate set of models that describes interesting behaviors without creating an overly large search space. Domain-specific reasoning is also important in the *test* phase, since models should be tested using abstract, high-level reasoning whenever possible. The information on which the generate and test phases are based, however, is highly heterogeneous, varying greatly in utility, applicability, and form, and it can be very difficult to tailor the reasoning level to the task. More-restrictive domains tend to admit more-powerful analysis tools and have much smaller search spaces. In linear mechanical systems, for instance, the *impulse response* shows the natural resonant and anti-resonant frequencies as spikes, and the *mode shapes* between those spikes show whether a vibrating mechanical system is mass- or stiffness-dominated[14]. In viscoelastics, an even more restrictive domain, three qualitative properties of a "strain test" can be exploited to reduce the search space of models to linear[10]. Assessing when and how to apply these types of tools is a non-trivial knowledge representation and reasoning problem.

Our term for a set of representation and reasoning tools for automatic model construction and testing is a *model-building domain*, which contains:

- A set of prototypical domain components
- A model generator—a function that combines components into candidate

models

- A set of data analysis tools that create qualitative and quantitative knowledge appropriate to the system at hand
- A set of rules and associated reasoning modes that apply this knowledge to discover inconsistencies between the candidate model and the unknown system.

Model building domains are similar in that they begin with a set of components and return a candidate ODE model; they differ not only in their generality (and thus the size of their search spaces) but also in the type and utility of their analysis tools, reasoning modes, and rules.

Creating an individual model-building domain from scratch for each specific application is inefficient,however, since many of the representations and reasoning tools involved in model generation apply to multiple domains. Rather, one wants a *hierarchy* of domains, where the specific domains can inherit the structure of their more-general parents. Our hierarchy is designed so that more-general domains, such as `linear-systems`, can be easily customized into more-specific ones, such as `linear-mechanical-systems`. Analysis tools may also exploit such a hierarchy; nonlinear time-series analysis, for example, can reveal the lower bound on the dimensionality of any system of ODEs, whereas a step response test is only useful in modeling linear systems, and a creep test is even more domain-specific. The basic element of the hierarchy is a *meta-domain*: a general framework that arranges components into models by relying on modeling techniques that transcend individual application domains. This paper introduces two such meta-domains: `linear-plus` and `xmission-line`. The `linear-plus` meta-domain takes advantage of fundamental linear-systems properties that allow the linear and nonlinear components to be treated separately under certain circumstances, which dramatically reduces the model search space. The `xmission-line` meta-domain generalizes the notion of building models using an iterative pattern, similar to a standard model of a transmission line, which is useful in modeling distributed systems. Meta-domains can be customized into more-specific domains or used directly; we demonstrate both approaches in the following section. We chose this particular pair of meta-domains as a good initial set because they cover a wide variety of engineering domains; we are exploring other possible meta-domains, especially for the purpose of modeling larger nonlinear networks.

Designing a representation that was flexible and powerful enough to support model-building in this domain/metadomain framework is a nontrivial problem. Our solution is based on an energy-based modeling representation called generalized physical networks (GPNs) which are similar to bond graphs[15] in that they maintain conservation of energy through *flow* and *effort* state variables but are easier to use as causality issues are not modeled. GPNs are general and powerful enough to represent models in a wide variety of application areas, and they adapt easily to different amounts of domain knowledge. In particular, they bring out similarities between components and properties in different domains via generalized components such as `linear-proportional`[1], which models energy dissipation. An abstract component may be customized—into `linear-resistor`, `linear-damper`, etc.—depending upon the application. In this manner, the available domain knowledge selectively sharpens the model in *appropriate* and *useful* ways.

3 Applying Meta-domains in PRET

PRET[5, 7] is an automatic SID tool that constructs ordinary differential equation (ODE) models of lumped-parameter continuous-time nonlinear dynamic systems. It takes a generate-and-test approach, using a small, powerful domain theory to build models, and then uses a body of mathematical and physical knowledge encoded in first-order logic to test those candidate ODEs against behavioral observations of the target system. Unlike other AI modeling tools—most of which use libraries to build models of small, well-posed problems in limited domains—PRET builds models of nonlinear systems in multiple domains and uses sensors and actuators to interact directly and automatically with the target system.

PRET relies heavily on the notions of model-building domains and meta-domains that were introduced in the previous section. It currently incorporates five specific GPN-based modeling domains: `mechanics`, `viscoelastics`, `linear-electronic`, `linear-rotational`, and `linear-mechanical`. These domains are dynamic: if a domain does not contain a successful model, it automatically expands to include additional components. For example if PRET fails to find a model in the `linear-electronics` domain, whose basic components include {`linear-resistor`, `linear-capacitor`}, that domain automatically expands to include `linear-inductor`. If a user wants to apply PRET to a system that does not fall in an existing domain, he or she can either build one from scratch—a matter of making a list of components and connectors—or use one of PRET's meta-domains. These meta-domains, which currently include `linear-plus` and `xmission-line`, use modeling techniques that transcend individual application domains: linear system fundamentals in the former and an iterative structure in the second.

PRET has successfully been applied to a variety of system identification tasks, ranging from textbook engineering problems to a radio-controlled car used in a deployed robotics system[6]. This section presents two examples that demonstrate how domains and meta-domains contribute to this process.

3.1 Modeling a Shock Absorber

Hydraulic shock absorbers, common in modern commercial vehicles, are complex nonlinear devices whose behavior depends upon the amplitude and frequency of the imposed motion. Accurate mathematical models of this behavior are

[1] flow is linearly proportional to effort

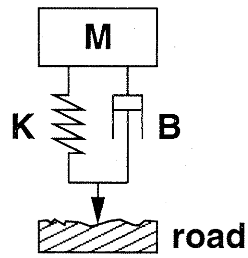

Figure 2: A one-degree-of-freedom quarter-vehicle model that includes a shock absorber—a viscous damping element, B—connected in parallel with a spring, K, and the loading effects of the car, M.

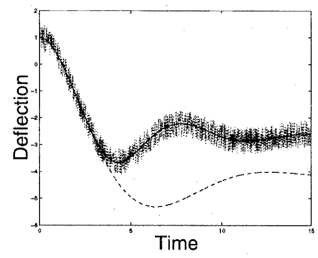

Figure 3: Step responses of (a) a hydraulic shock absorber (dotted) (b) an unsuccessful candidate model with a linear spring (dashed) and (c) a successful model, which incorporates a cubic spring (solid).

key to realistic vehicle simulations and active-suspension controllers. Shock-absorber models normally come in two forms: either as a stand-alone shock absorber—typically just a connected spring and damper element—or as part of a quarter-vehicle model, where the loading effects of one quarter of the vehicle are included. The effects of different shock absorbers in one- and two-degree-of-freedom quarter vehicle models may be found in [17]; the behavior of five damper model variants, such as "no spring," "velocity-dependent damping," or "no linear spring," is described by [3]. This is exactly the kind of expert reasoning that motivated the design of the domain knowledge framework described here; these kinds of similarities make it easy for engineers to use PRET on real problems.

The user initializes PRET on a modeling problem through a `find-model` call, a fragment of which is shown in Figure 4. The first line specifies the domain (`linear-mechanics`) which instantiates the relevant domain theory; the next informs PRET of the relationship between state variables. `hypotheses` and `drives` are optional lists of model fragments describing different possible aspects of the physics involved in the system, e.g. a spring force that obeys a cubic version of Hooke's law, and a constant drive term that represents a constant normal force on the suspension. `observations` are facts about the system (which are assumed to be true) and may be measured automatically by sensor and/or interpreted by the user. In this case, the `observations` are a noisy

```
(find-model
  (domain linear-mechanics)
  (state-variables (<x> (integral <v>)))
  (hypotheses
    ((<force> (* k (cube (integral <v>)))))
  (drive (<force> d))
  (observations
    (numeric (<time> <x>)
             ((0 1.3) (0.1 1.2) ...))
  (specifications
    (<x> absolute-resolution 1.2 (0 15))))
```

Figure 4: A `find-model` call for the shock absorber. The state variable `<v>` is velocity (i.e. $v = \frac{dx}{dt}$, where $x =$ `<x>` is the deflection from the equilibrium position). The hypothesis represents a cubic spring force: $F = kx^3$. By default, the `linear-mechanics` domain includes linear damping, inertia, and spring components, so PRET's user need not specify them explicitly. The `numeric` observation is the noisy dotted time series shown in the previous figure.

time-series measurement of the deflection (shown graphically in Figure 3(a)). By default, the `linear-mechanics` domain includes linear damping, inertia, and spring components, so the user need not enter them explicitly; PRET's model generator will automatically include them along with the user-specified hypotheses. Geometric pre-processing of the numerical observation shows that the state variable `<x>` undergoes a damped oscillation to a fixed point. From these facts, PRET's model tester[19] deduces (among other things) that the order of any linear model must be at least two. This qualitative information lets PRET immediately rule out the first dozen or so candidate models. Proceeding to slightly more complex ODEs, PRET generates the model $a\ddot{x} + b\dot{x} + cx + d = 0$, which is made up of three built-in domain hypotheses and the user's drive hypothesis. Its model tester cannot rule out this model using qualitative reasoning techniques, and so is forced to invoke its nonlinear parameter estimator to establish that the solution to this ODE, shown in Figure 3(b), does not match the numeric observation shown in Figure 3(a). After discarding a variety of other unsuccessful candidate models by various means, PRET eventually generates and tests the ODE $a\ddot{x} + b\dot{x} + cx + kx^3 + d = 0$. This model passes all qualitative and quantitative checks, and so is returned as PRET's output:

```
 ... no refutation ...
(model ((= (+ (* (const a) (deriv (deriv <x>)))
              (* (const b) (deriv <x>))
              (* (const c) <x>)
              (* (const k) <x> <x> <x>)
              (const d)) 0)
 ((a 1.00) (b 0.50) (c 0.31) (k 0.024) (d 1.30))))
```

The `linear-mechanics` domain can be implemented in several ways. The easiest is to use the `linear-plus` meta-domain and add several domain-specific components: `linear-mass`, `linear-damping` and `linear-spring`. These are just the generalized components `linear-capacitance`, `linear-resistance` and `linear-inertia`—renamed in a manner that makes their meaning obvious to someone who would be using the `linear-mechanics` domain. Jargon matching is only a small part of the power of meta-domain customization, however; domain knowledge allows PRET to selectively sharpen its knowledge. If PRET knows that the system is linear and mechanical, for instance, the general component `linear-inertia` takes on the more-specific meaning associated with `linear-spring`, such as the knowledge that mechanical springs often have appreciable mass and internal friction that cannot be neglected.

Implementing the `linear-mechanics` domain using the `linear-plus` meta-domain has another very important advantage for problems like this, which have a few drive terms and a few nonlinear terms. `linear-plus` separates components into a linear and a non-linear set in order to take advantage of two fundamental properties of linear systems: (1) there are a polynomial number of unique nth-order linear ODEs[9], and (2) linear system inputs (drive terms) appear verbatim in the resulting ODE system. The first of these properties converts an otherwise-exponential search space to polynomial; functionally equivalent linear networks reduce to the same Laplace transform transfer function, which allows PRET to identify and rule out any ODEs that are equivalent to models that have already failed the test. The second property allows this meta-domain (and thus any specific domain constructed upon it) to handle a limited number of nonlinear terms by treating them as system inputs. As long as the number of nonlinear hypotheses remains small, the search space of possible models remains tractable.

PRET's user could also skip the `linear-mechanics` domain and use the `linear-plus` meta-domain directly for this problem, simply by specifying a few extra terms in the `hypotheses` line of the `find-model` call (e.g., `(<force> (* k (integral <v>)))` and so on). The only difference between doing this and using the `linear-plus` meta-domain with extra components would manifest in PRET's run time, as it would no longer be able to use domain knowledge to streamline the generate and test phases. Indeed, one could even omit *all* hypotheses, since PRET automatically performs power-series expansions if it runs out of user and domain hypotheses, but that would increase the run time even further. The best course of action is to use as much domain knowledge as possible, and PRET's layered domain/meta-domain framework is designed to make it easy to do so.

3.2 Water Resource Systems

Water resource systems are made up of streams, dams, reservoirs, wells, aquifers, etc. In order to design, build, and/or manage these systems, engineers must model the relationships between the inputs (e.g., rainfall), the state variables (e.g., reservoir levels), and the outputs (e.g., the flow to some farmer's irrigation ditch). To do this in a truly accurate fashion requires partial differential equations (PDEs) because the physics of fluids involves multiple independent variables—not just time—and an infinite number of state variables. PDEs are extremely hard to work with, however, so the state of the art in the water re-

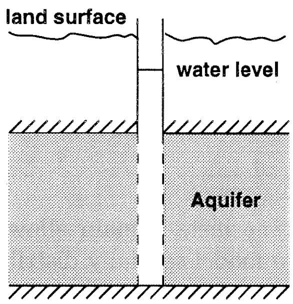

Figure 5: An idealized representation of an open well penetrating an artesian aquifer. The motion of the water level in the well is controlled by sinusoidal fluctuations of the pressure in the aquifer.

source engineering field falls far short of that. Most existing water resource applications, such as river-dam or well-water management systems, use rule-based or statistical models. ODE models, which capture the dynamics more accurately than these simple models but are not as difficult to handle as PDEs, are a good compromise between these two extremes, and the water resource community has begun to take this approach[8, 11]. In this section, we show how meta-domains help PRET duplicate some of these research results and model the effects of sinusoidal pressure fluctuation in an aquifer on the level of water in a well that penetrates that aquifer. See Figure 5 for a schematic. This example is a particularly good demonstration of how domain knowledge and the structure inherited from the meta-domain let the model generator build systems without creating an overwhelming number of models. This example also demonstrates how GPNs allow PRET to model a variety of systems using the same underlying representation and to incorporate the load effects easily and naturally into the model.

The first step in describing this modeling problem to PRET is to specify a domain. Because there is no built-in "water resource" domain, the user would have to choose (and perhaps customize) one of the meta-domains. For this problem, the choice is obvious, as the `xmission-line` meta-domain is specifically designed for this kind of *distributed* physics, which turns up in fluid flows, vibrating strings, gas acoustics, thermal conduction and diffusion, etc.[13]. This meta-domain serves as a bridge between two very different paradigms. GPN components represent prototypical *lumped* elements, each of which models a single physical component, whereas a system like a transmission line or a guitar string can be thought of as an infinite number of small elements (the basis of a PDE model). Using the former to model the latter requires an incremental approach. In particular, one can approximate a spatiotemporally distributed system using an iterative structure with a large number of identical lumped sections, each of which corresponds to an ODE term. Figure 6 shows a diagram of this: a generalized iterative two-port network with n sections, each of which has a serial (A_i) and a parallel (B_i) component. In a basic model of an elec-

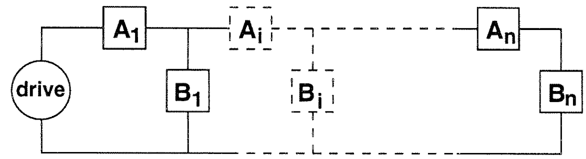

Figure 6: The `xmission-line` meta-domain allows PRET to use its lumped-element GPN components to model spatially distributed systems.

```
(find-model
  (domain xmission-line)
  (state-variables (<well-flow> <flow>)) ...
  (hypotheses
    (<effort> (* c (integral <flow>)))
    (<effort> (* l (deriv <flow>)))
    (<effort> (* r <flow>)))
  (drive (<effort> (* da (sin (* df <time>)))))
  (observations
    (not-constant <well-flow>)
    (not-constant (deriv <well-flow>))
    (numeric (<time> <well-flow> (deriv <well-flow>))
     ((0 4.0 0.5) (0.1 4.25 0.6) ...)))  ...)
```

Figure 7: A `find-model` call fragment for the well/aquifer example of Figure 5.

trical transmission line, for example—whence the name of the `xmission-line` meta-domain—typical electrical parameters, such as resistance or inductance, are given in per-unit-length form, and the A_i and B_i would correspond to resistors and capacitors, respectively.

As in the shock absorber example, PRET's user can either customize the meta-domain or use it directly. For the well/aquifer problem, this customization would consist of renaming the general components `linear-capacitance`, `-inertia`, and `-resistance` to match the standard domain vocabulary; capacitive and resistive effects, in particular, simulate radial flow in an aquifer, and water mass is treated as inertia. (These concepts and equivalences, which appear in every textbook, are a routine part of a water-resource practitioner's knowledge.) The domain could be further customized based upon knowledge of the aquifer's forcing function; if the water's velocity changes slowly, for example, inertia effects can normally be ignored. For the purposes of demonstrating how one uses a meta-domain directly, however, we omit this customization in this example, so the `find-model` call of Figure 7 simply instantiates the `xmission-line` meta-domain.

This call differs from the previous examples in a variety of ways. This meta-domain has no built-in components; it only provides the template of an iterative network structure. PRET must therefore rely solely on user-specified hypotheses

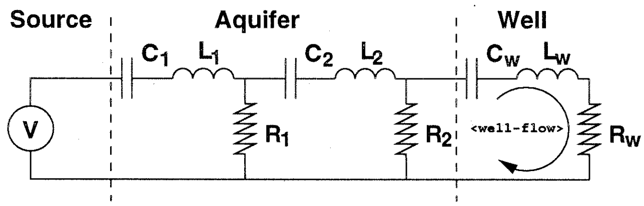

Figure 8: PRET's model of the well/aquifer system. The drive, $V = d_a \sin d_f t$, simulates a sinusoidal pressure fluctuation in the aquifer; the arrow labeled `<well-flow>` in the network corresponds to the water flow into and out of the well. Note how the `xmission-line` meta-domain and GPN components naturally incorporate the load (the well) into the model.

to build models[2]. State variables in hypotheses bear domain-independent names like `<effort>` and `<flow>`, rather than domain-specific ones like `<force>`, or `<water-flow>`. The `xmission-line` meta-domain builds models with an iterative structure, using these `<flow>` and `<effort>` hypotheses as the parallel and series components of the sections and dynamically creating instances of each kind of state variable as segments are added to the model. Since numeric observations describe specific state variables (e.g. the numerical observation of `<well-flow>` in the `find-model` call), their identifiers are prespecified in the `state-variables` line of the call. Finally, unlike the shock absorber, the well/aquifer includes a *nonautonomous* drive term—one that has an explicit time dependence.

As before, PRET automatically searches the space of possible models, using the `xmission-line` meta-domain template to build models and qualitative and quantitative techniques to test them, until a successful model is found. The result is shown in Figure 8. A perfect model of an infinite-dimensional PDE requires an infinite number of discrete sections, but one can construct approximations using only a few sections, and the fidelity of the match rises with the number of sections[3]. In this case, PRET used two `xmission-line` sections to model the aquifer and one to model the well. This incorporation of the well as an integral part of the model is an important feature of the model-building framework described in this paper. Finally, like `linear-plus`, the `xmission-line` meta-domain lets PRET avoid duplication of effort. Connecting arbitrary components in parallel and series creates an exponential number of models, many of which are mathematically equivalent (cf., Thévenin and Norton equivalents, in network theory). The `xmission-line` meta-domain avoids this duplication by first limiting the number of possible component combinations in the initial network model (A_1 and B_1 of Figure 6), and then incrementing this structure

[2]In other domains, PRET uses power-series expansions if it runs out of user hypotheses. Since the basic paradigm in `xmission-line` is essentially a spatial expansion, a power-series expansions would be a duplication of effort.

[3]Hence the notion of an ODE truncation of a PDE, which is exactly what PRET is constructing here.

to a limited depth before attempting another initial network.

4 Related Work

Much of the pioneering work in the qualitative reasoning (QR) modeling community focuses on reasoning about pre-existing models: simulating them[16], simplifying and refining them[20], or keeping track of which model is appropriate in which regime[1]. QR model *construction* research has focused on building models from fragments[4, 12]. PRET uses some of the same techniques but has different goals and a different overall approach: it works with noisy, incomplete sensor data from real-world systems and attempts not to "discover" the underlying physics, but rather to find the simplest ODE that accounts for the given observations.

In the QR research that is most closely related to PRET's domains and meta-domains, ODE models are built by evaluating time series using qualitative reasoning techniques and then applying parameter estimation to match the resulting model with a given observed system[10]. This approach differs from the techniques presented in this paper in that it selects models from a set of pre-enumerated solutions in a very specific domain (linear viscoelastics). Amsterdam's automated model construction tool[2] uses a similar underlying component representation (bond graphs) and is applicable to multiple domains. However, it is also somewhat limited; it can only model linear systems of order two or less. The domain/meta-domain framework described in this paper is much more general; it works on linear *and nonlinear* lumped-parameter continuous-time ODEs in a variety of domains, and it uses *dynamic* model generation to handle arbitrary devices and connection topologies.

5 Conclusion

Model-building domains and meta-domains, coupled with generalized physical networks and a hierarchy of qualitative and quantitative reasoning tools that relate observed physical behavior and model form, provide the flexibility required for gray-box modeling of nonlinear dynamical systems. The two meta-domains introduced in this paper, `linear-plus` and `xmission-line`, use modeling techniques that transcend individual application domains to create rich and yet tractable model search spaces. This framework is flexible as well as powerful; one can use meta-domains directly or customize them to fit a variety of engineering applications. Both domains and meta-domains adapt smoothly to varying amounts and levels of domain knowledge.

References

[1] S. Addanki, R. Cremonini, and J. Penberthy. Graphs of models. *Artificial Intelligence*, 51:145–178, 1991.

[2] J. Amsterdam. *Automated Qualitative Modeling of Dynamic Physical Systems.* PhD thesis, MIT, 1992.

[3] F. Besinger, D. Cebon, and D. Cole. Damper models for heavy vehicle ride dynamics. *Vehicle System Dynamics*, 24:35–64, 1997.

[4] D. Bobrow, B. Falkenhainer, A. Farquhar, R. Fikes, K. Forbus, T. Gruber, Y. Iwasaki, and B. Kuipers. A compositional modeling language. In *QR-96*, 1996.

[5] E. Bradley, M. Easley, and R. Stolle. Reasoning about nonlinear system identification. Technical Report CU-CS-894-99, University of Colorado at Boulder, 2000. In review for *Artificial Intelligence.*

[6] E. Bradley, A. O'Gallagher, and J. Rogers. Global solutions for nonlinear systems using qualitative reasoning. *Annals of Mathematics and Artificial Intelligence*, 23:211–228, 1998.

[7] E. Bradley and R. Stolle. Automatic construction of accurate models of physical systems. *Annals of Mathematics and Artificial Intelligence*, 17:1–28, 1996.

[8] J. Bredehoeft, H. Cooper, and I. Papadopulos. Inertial and storage effects in well-aquifer systems. *Water Resource Research*, 2:697–707, 1966.

[9] W. Brogan. *Modern Control Theory.* Prentice-Hall, New Jersey, 3rd edition, 1991.

[10] A. Capelo, L. Ironi, and S. Tentoni. Automated mathematical modeling from experimental data: An application to material science. *IEEE Transactions on Systems, Man and Cybernetics - C*, 28:356–370, 1998.

[11] D. Chin. *Water-Resource Engineering.* Prentice Hall, New Jersey, 2000.

[12] B. Falkenhainer and K. Forbus. Compositional modeling: Finding the right model for the job. *Artificial Intelligence*, 51:95–143, 1991.

[13] M. Ghausi and J. Kelly. *Introduction to Distributed-Parameter Networks.* Holt, 1968.

[14] J.-N. Juang. *Applied System Identification.* Prentice Hall, Englewood Cliffs, N.J., 1994.

[15] D. Karnopp, D. Margolis, and R. Rosenberg. *System Dynamics: A Unified Approach.* Wiley, New York, second edition, 1990.

[16] B. Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:289–338, 1986.

[17] R. Langlois and R. Anderson. Preview control algorithms for the active suspension of an off-road vehicle. *Vehicle System Dynamics*, 24:65–97, 1997.

[18] R. Sanford. *Physical Networks.* Prentice-Hall, 1965.

[19] R. Stolle and E. Bradley. Multimodal reasoning for automatic model construction. In *AAAI-98*, 1998.

[20] D. Weld. Reasoning about model accuracy. *Artificial Intelligence*, 56:255–300, 1992.

[21] D. Weld and J. de Kleer, editors. *Readings in Qualitative Reasoning About Physical Systems.* Morgan Kaufmann, San Mateo CA, 1990.